テーマ１：指数関数時間アルゴリズム

本日の講義の目的

- NP-困難問題（最大独立点集合問題）の指数時間アルゴリズム

テーマ１では，指数時間かかる（と思われている）問題に対する指数時間のアルゴリズムの改良について議論する．こうしたアルゴリズムは，近似を目標としたり，平均的に解くことを目標としているアルゴリズムと区別して，厳密解アルゴリズム (exact algorithm) と呼ばれている [1].

1.1. NP-問題，NP-型問題と指数時間アルゴリズム

一般に NP-問題，NP-型問題（たとえば NP-型最適化問題）に対しては，単純な指数時間アルゴリズムを設計することができる．

定理 1.1. 問題 X を任意の NP-問題，もしくは NP-型最適問題とする．また，そのサイズパラメータを n とする．このとき，適当な多項式 $p_X(n)$ に対し，X を $O(2^{p(n)})$-時間で解くアルゴリズムを構成することができる．

例で考えてみる．たとえば，3SAT 問題，ハミルトン閉路問題 (HAM)，点彩色問題 (COLOR) に対して，それを解く指数関数時間アルゴリズムを考えることは容易だろう．ただし，指数関数の大きさは，次のように各々かなり異なる．

問題名	サイズ n の意味	単純なアルゴリズムの計算量
3SAT	変数の個数	$O(2^n)$
HAM	頂点数	$O(n^n) = O(2^{n\log n})$
COLORING	頂点数	$O(n^K) < O(n^n) = O(2^{n\log n})$

（復習）

点彩色問題（判定版）(COLOR)
入力： 無向グラフ $G = (V, E)$, 整数 K.
仕事： G は K 色（以下）で頂点彩色が可能か？

こうした単純な指数時間アルゴリズムに対して，同じ指数関数でも，もっと低い指数関数のアルゴリズムは設計できないだろうか？厳密解アルゴリズムの研究の目標は，そうした改善をできるかぎり得ることである．テーマ１では，そのために開発された技法を見ていくことにしよう．

1.2. 最大独立点集合問題に対する厳密解アルゴリズム

まずは，準備運動として最大独立点集合問題 (MIS) に対するアルゴリズムを紹介する．

（復習）

最大独立点集合問題（判定版） (Max. Independent Set problem, MIS)
入力： 無向グラフ $G = (V, E)$，整数 M
仕事： G は K 頂点（以上）の独立点集合を持つか？

頂点被覆問題（判定版） (Vertex Cover problem, VC)
入力： 無向グラフ $G = (V, E)$, 整数 M.
仕事： G は K 頂点（以下）の頂点被覆を持つか？

定理 1.2. 3SAT $\leq^{\mathrm{poly}}$ VC $\leq^{\mathrm{poly}}$ MIS.

したがって，MIS は NP-完全な問題の一つである．一方，MIS に対しては $O(2^n)$-時間アルゴリズムを容易に作ることができる．それを改善する方法を示したのが次の定理だ.

定理 1.3. MIS は $\mathcal{O}^*\left(3^{n/3}\right)$-時間計算可能.
補足． (1) $3^{1/3} = 1.442\cdots = 2^{0.52\cdots}$.
(2) 記法 $\mathcal{O}^*(e(n))$ は「適当な定数 $c > 0$ に対して $cn^c e(n)$ 以下である」という意味.

証明： ここでは簡単のため，the maximum independent set の要素数を求める問題を MIS と呼ぶことにする．以下のアルゴリズムが MIS を解くアルゴリズムである．

Algorithm MIS#1
　input: $G = (V, E)$;
　output: the size of the maximum independent set;
begin
　if then$|V| = 0$ **thenreturn** 0;
　v = one vertex of min. degree in G;
　return $1 + \max\{\,\mathrm{MIS\#1}(G - \mathrm{N}[y]) \mid y \in \mathrm{N}[v]\}$;
　注）$\mathrm{N}[v]$ は v と v に隣接する頂点の集合.
end.

このアルゴリズムは再帰的に定義されているので，その正当性や 最悪時間計算量 の解析は帰納的に行うとスムースにできる．以下では最悪時間計算量の解析を示す．

ここでは MIS#1 の呼び出し回数に基づく計算量を解析する．そのために，関数 $T(n)$ を，n 頂点のグラフ G で最も呼び出し回数が多い場合の呼び出し回数とする．すると，帰納的に以下の漸化式が得られる．

$$
\begin{aligned}
T(n) &\leq 1 + \sum_{y \in \mathrm{N}[v]} T(n - |\mathrm{N}[y]|) \\
&\leq 1 + (d(v) + 1) T(n - d(v) - 1).
\end{aligned}
$$

ここで，$d(v)$ は頂点 v の次数とした．つまり $|\mathrm{N}[v]|$ である．また，$n' \leq n''$ のとき，$T(n') \leq T(n'')$ であることを利用した．

最小次数は計算が進んでも（つまり，再帰が進んでも）非減少である．したがって，$s = d(v) + 1$ に固定して考えた場合が上界となる．そこで

$$T(n) \ \leq \ 1 + sT(n-s) \ \leq \ 1 + s + s^2 + s^3 + \cdots + s^{n/s} \ = \ \frac{1 - s^{n/s}}{1 - s} \ \leq \ s^{n/s} \ = \ \left(s^{1/s}\right)^n$$

という上界式が得られる．ここで，関数 $s^{1/s}$ は $s = \mathrm{e}$ で最小値をとるが，整数では $s - 3$ で最小となることを用いれば，上界 $3^{n/3}$ が得られる． □

この証明での上界は特殊な場合の上界（つまり，本当はあまり起こりそうのない上界）を使っている．もっと詳しい解析を用いれば上界を改良する余地はある [1]．また，実際の計算時間も，この上界よりははるかに速いはずである．

ここに示したような再帰をうまく使って解く方法は，指数関数計算時間を削減する代表的な手法である．この方法では，(i) 領域計算量が多項式ですむ場合が多い，(ii) 列挙にも使うことができる，などの利点がある．

参考文献

以下の本は厳密解指数時間アルゴリズムを網羅的に解説した本．このテーマで紹介するアルゴリズムや解析はこの本に基づくものである．

[1] F. Fomin and D. Kratsch, *Exact Exponential Algorithms*, Springer, 2010.